AI 2020

HACK-A-THRON

Gagik Gavalian (Jlab)

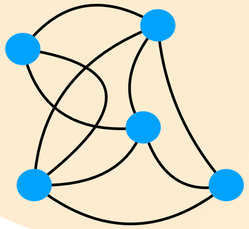
TEAM: OXPECKER

Will Phelps (CNU)

Alexandr Austregesilo (JLAB)

Tyler Viducic (ODU)

Gagik Gavalian (JLAB)



Software

Machine Learning (DL,AI) Software:

- **Python (Keras, Tensor-flow):**
  - Have to install bunch of packages (once installed and versions are matching things work).
  - Have to install on other systems to run the code.
  - Notebooks come with preinstalled packages.
  - Very nice short code.

- **DL4J (Java based software):**
  - Tensor-flow and Keras come compiled for native platforms
  - One dependency to include in the project (no hassle with many packages)
  - Once compiled can be deployed on any platform (Mac, Linux, Windows)
  - Works well in the notebook
  - Has additions to accelerate (CUDA drivers, again with native platforms and JNI)

Software

- Making Network Configurations are very easy
- Supports most of the networks used in python:
- MultiLayer Perceptron (MLP)
- Recurrent Neural Networks (RNN,LSTM, GRU)
- Convolutional Neural Networks (CNN)

```
public MultiLayerConfiguration createConfiguration(){
      MultiLayerConfiguration conf;
      NeuralNetConfiguration.Builder builder
               = new NeuralNetConfiguration.Builder()
                    .optimizationAlgo(
                     OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
                    .updater(new Adam(0.001));
       NeuralNetConfiguration.ListBuilder listBuilder = builder.list();

       listBuilder.layer(new DenseLayer.Builder()
                  .nIn(nInputs).nOut(hiddenLayers[0])
                  .weightInit(WeightInit.XAVIER)
                  .activation(Activation.RELU).build());

       for(int i = 0; i < hiddenLayers.length - 1; i++){
           listBuilder.layer(new DenseLayer.Builder()
                      .nIn(hiddenLayers[i]).nOut(hiddenLayers[i+1])
                      .weightInit(WeightInit.XAVIER)
                      .activation(Activation.RELU).build());
       }

       listBuilder.layer(new OutputLayer.Builder(LossFunction.MSE)
                       .activation(Activation.RELU)
                       .nIn(hiddenLayers[hiddenLayers.length - 1])
                       .nOut(nOutputs).build());
       return listBuilder.build();
   }
```

Software

- Writing Training scripts (programs) are very easy as well
- Java has several options for writing scripts:
  - natural scripting language (called Groovy)
  - Jython, it's python with Java classes imports
  - since JDK 9, JShell, which is very similar to CINT


```
ChallangeDataReader reader = new ChallangeDataReader();
INDArray input  = reader.readInput("TRAIN/TRAIN.csv", 100000);
INDArray output = reader.readOutput("TRAIN/TRAIN_labels.csv", 100000);

DataSet ds = new DataSet(input,output);
MLPRegression reg = new MLPRegression( 3600, 1, new int[]{24 , 24, 24});
reg.train(ds, null, 20); // run for 20 epochs

INDArray evalInput = reader.readInput("TEST/TEST.csv", 1000);

reg.eval(evalInput);
```

Neural Network (Problem 1) (Tyler)

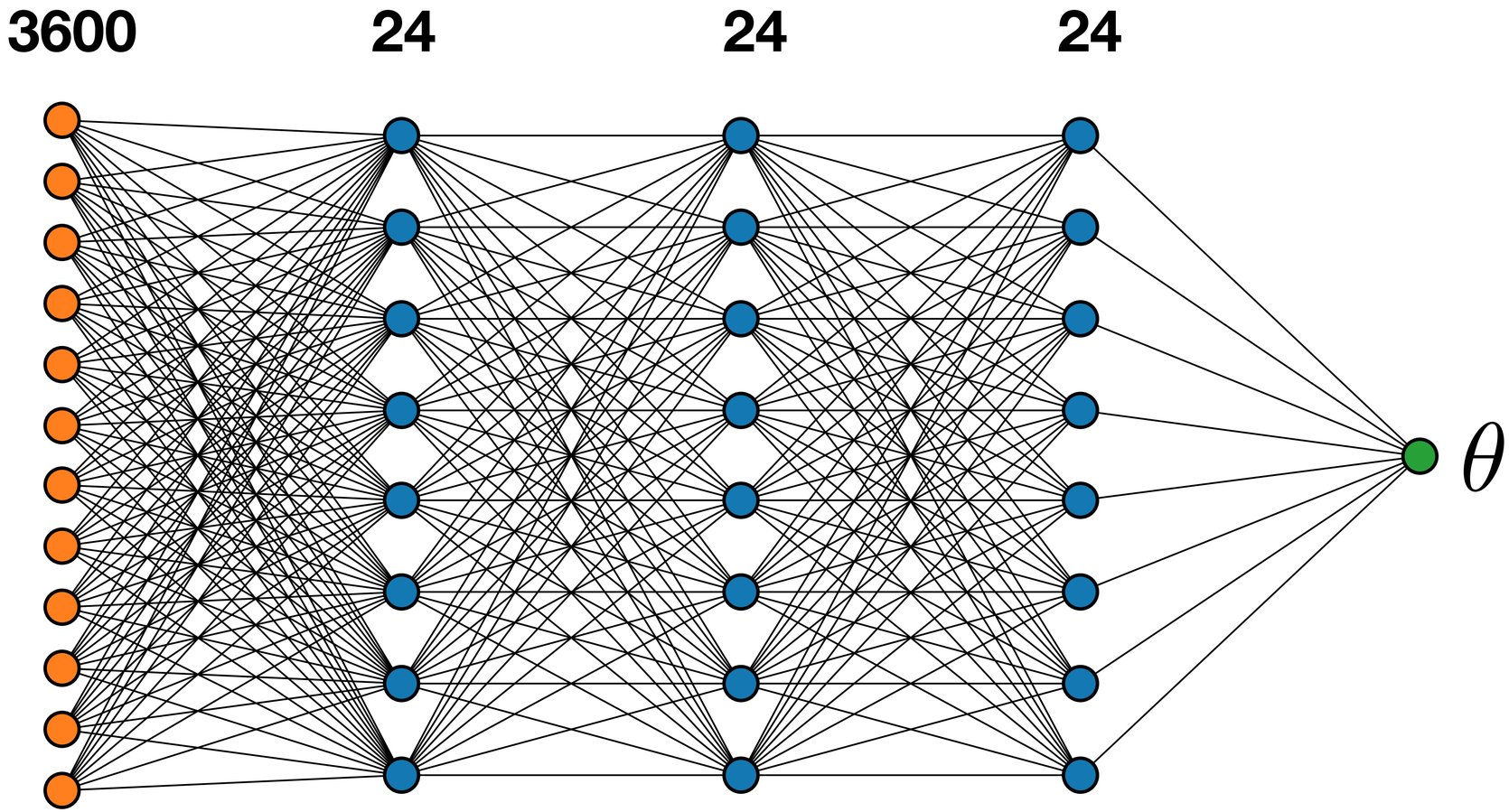
- **Solution:**
  - Passed dataframe to sklearn LinerarRegression class
  - For results not in range (-10,10) - replace with results from NN solution

```
dataset = pd.read_csv('/media/tylerviducic/Elements/aiHack/Set1/TRAIN/TRAIN.csv')
X = dataset.iloc[:25000, :].values
solutions = pd.read_csv('/media/tylerviducic/Elements/aiHack/Set1/TRAIN/TRAIN_labels.csv')
y = solutions.iloc[:25000, 1].values
test = pd.read_csv('/media/tylerviducic/Elements/aiHack/Set1/TEST/TEST.csv')
x_test = test.iloc[:, :].values

# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X, y)
```

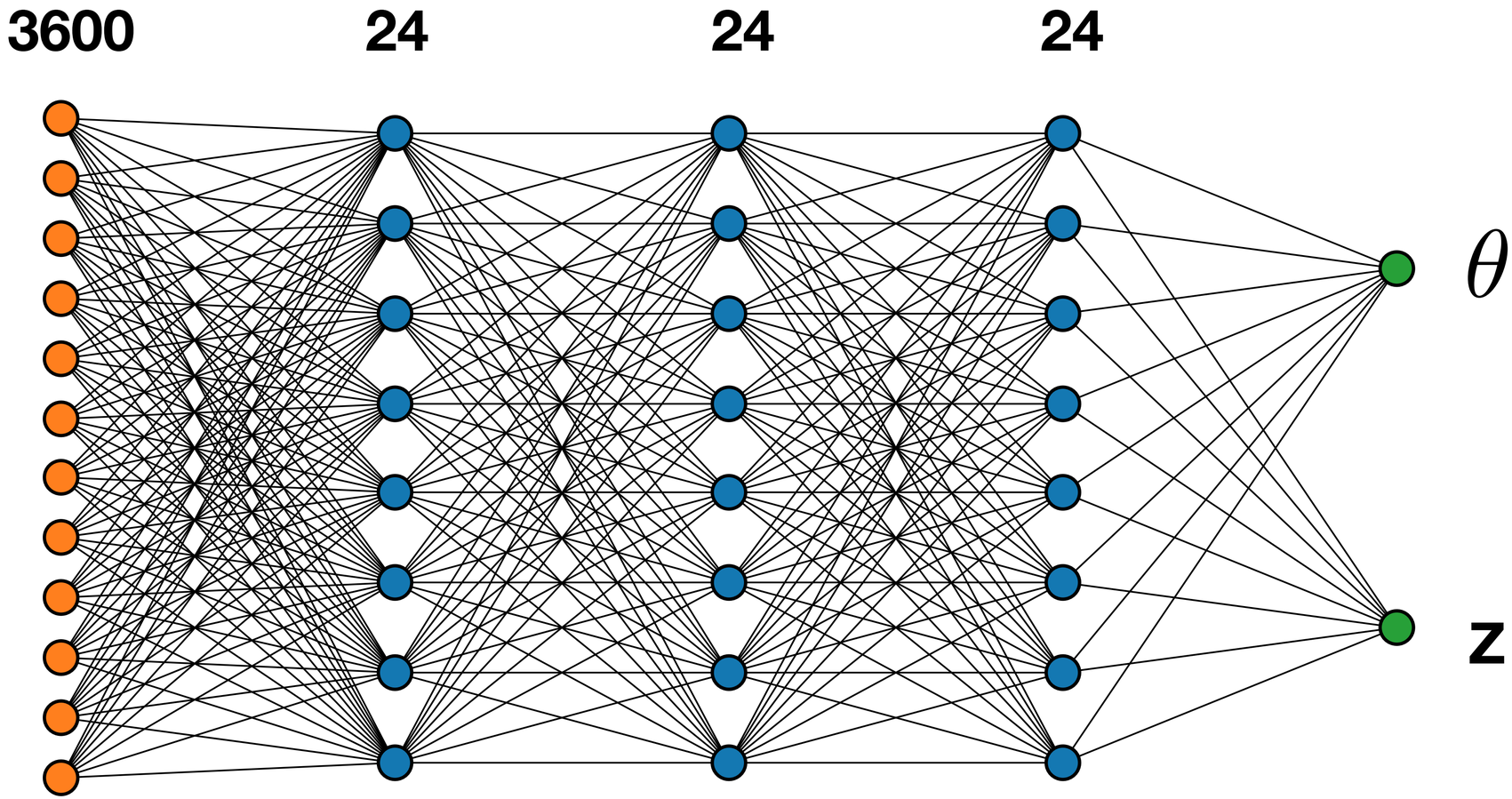
Neural Network (Problem 2)

- **Input:**
  - 3600 nodes represents all pixels in the image.
  - Three hidden layers (24 nodes each)
- **Output**
  - Angle of the track
- **Training:**
  - Could not train for a long time due to number of parameters.
  - Couldn't increase number of nodes in the hidden layers due to training speed.



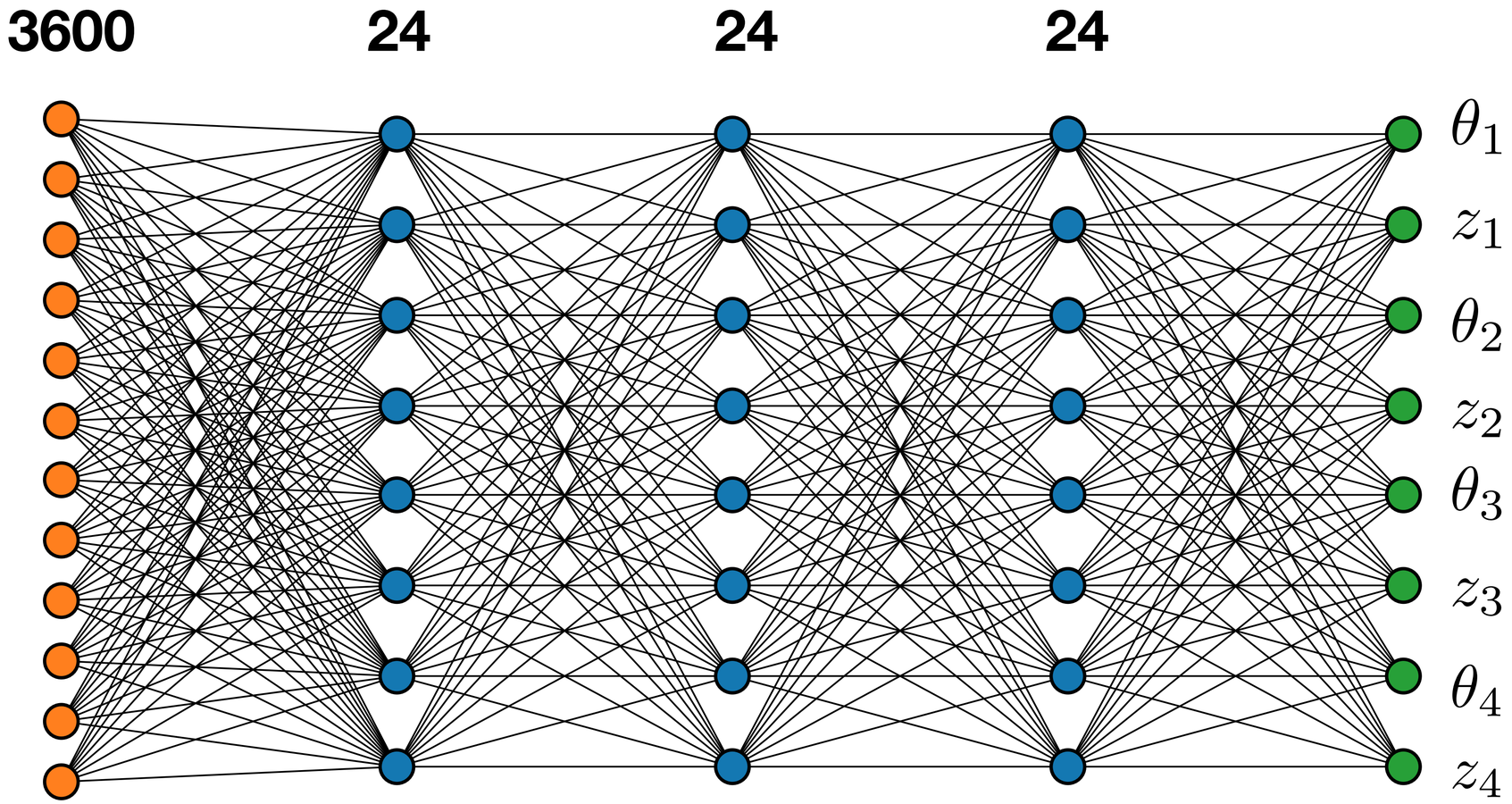
Neural Network (3&4)

- **Input:**
  - 3600 nodes represents all pixels in the image.
  - Three hidden layers (24 nodes each)
- **Output**
  - Angle of the track
  - Z vertex of the track
- **Training:**
  - Could not train for a long time due to number of parameters.
  - Couldn't increase number of nodes in the hidden layers due to training speed.



Neural Network (5)

- **Input:**
  - 3600 nodes represents all pixels in the image.
  - Three hidden layers (24 nodes each)
- **Output (8 nodes)**
  - 4 angles of tracks
  - 4 vertices of tracks
- **Surprises:**
  - It did better than expected and was best solution.
  - It would make more sense to separate clusters of hits into separate inputs and infer each track separately using solution for 3rd problem.



Problems #2 & #3 (Will)

- Keras w/TensorFlow Backend
- Spent a lot of time getting environment working again before starting (Could have used AWS)
- Eventually used JLab resources for training
  - Python scripts running on JLab's Nvidia Titan RTX cards


```
test = pd.read_csv("TRAIN/TRAIN.csv")
labels = pd.read_csv("TRAIN/TRAIN_labels.csv")
activation = 'relu'

model = Sequential()
model.add(Dense(units=1000, activation=activation, input_shape=(3600, )))
model.add(Dense(units=1000, activation=activation))
model.add(Dense(units=1000, activation=activation))
model.add(Dense(units=2))
model.compile(optimizer=adam(lr=.001), loss='mean_squared_error', metrics=['accuracy'])

model.fit(test, labels[labels.columns[1:]], epochs=300, batch_size=256, validation_split=0.2)
```

Lessons Learned

- Never trust Thomas (*Never*)
  - The training and testing samples were contaminated.
- Check the input (training) data.
- Prepare the working environment ahead of time:
  - I assumed that this will be using your laptops kind of challenge
- Do not compete within the team.
  - At some point we were trying to beat each others scores.
  - Better to split the assignment and attack them in parallel

Summary

- **General Comments:**
  - It was very well organized and everyone had fun (as far as I could tell)
  - This kind of challenge can be done more often involving beginners to boost their enthusiasm.
  - Teams can be organized in a way to contain at least one “expert”

- **Suggestions (mild):**
  - Introduce participants to available environments ahead of time.
  - Have more diverse datasets and problems so team members can contribute to different tasks.
    - Regression, classification and series prediction
  - Publicize data formats (though it may be standard to python users, some people may be using different tools).
  - **Provide more coffee. (*very important*)**